Table of Contents

Updates on Issues

Known system issues that we are working on

Under Investigation

Slow Response to the "ls -l" Command

***Problem**: Users have experienced a slow response to the Unix command ls -l, often waiting for minutes for the command to complete.*

Status: In Progress

Actions:

Updated 08.31.11 - NAS systems staff have deployed a second new filesystem (nobackupp2). In addition, they completed the transition of users to nobackupp1 on August 8th.

Updated 07.27.11 - NAS systems staff have deployed a new filesystem (nobackupp1) with a new RAID controller, which improves the IOPS performance. Users on nobackupp10 are being transitioned to the new filesystem. Users on other nobackup filesystems will be transitioned to similar, new RAID controllers.

Updated 06.28.11 - NAS systems staff are testing new RAID controllers, which will improve Input/Output Operations Per Second (IOPS) performance, and result in an improved stats-per-second rate.

Updated 05.26.11 - NAS systems staff began an immediate investigation, and determined the source of the problem was related to access of the metadata on the Lustre filesystem, and took the following actions:

- Implemented a change to preload metadata information into the buffer cache on all of the nobackup filesystems. This improved the stats-per-second rate.
- Evaluated the solid-state drives (SSDs) to improve performance of stat calls. There are two use cases for accomplishing this: one approach is to use SSDs for filesystem journals, the other is to use them for filesystem metadata. SSDs were deployed on nobackupp10 for the filesystem journal.
- Reduced the default stripe count to 1 and the default stripe size to 4 MB. This change resulted in fewer stat calls and, over time, improve the ls -l response. (Effective January 13, 2011).
- Developed a modified gnu tar command, /usr/local/bin/mtar, which is Lustre stripe aware and will create a tar file or extract files with an appropriate sized stripe count.

Tips:

Updated 08.02.11 - Since the default stripe count was set to 1 to improve overall performance of Lustre, users need to keep in mind that they sometimes should use larger stripe counts, particularly with large files.

- You can set the stripe count on a directory or create a file with a particular stripe count by using the command *lfs setstripe*.
- We recommend using the *mtar* command in place of *tar* whenever you create or extract from a tar file on Lustre, as *mtar* will automatically choose appropriate stripe counts.

See more information on Lustre striping in the articles Lustre Basics and Lustre Best Practices.

Background:

Previously, by default, the Lustre filesystem striped data across object storage targets (OSTs) in 1-MB chunks. So, a 4-MB file would be spread across four OSTs, resulting in four stat calls to get the size of that one file.

MPI Program Fails or Hangs

Problem: *MPI program fails or hangs due to network communication problems.*

Status: In Progress

Actions:

NAS system staff are monitoring for errors in the InfiniBand fabric, and replacing bad or unreliable cables when detected.

Tips:

If your MPI job aborts or hangs due to InfiniBand problems, your PBS output file will produce error messages similar to the below:

```
MPI ERROR: 14:34:10: rank 960: r199i0n2 IB board mlx4_0 port 1

           had fault with communications to r190i0n6, restarting...
```

In this case, we recommended that you do the following:

1. Wait a few minutes and resubmit your job.
2. File a ticket with the NAS Control Room to report the problem.
3. If you have not done so, use SGI's MPT versions 2.0.4 or later, which are more robust against these InfiniBand problems and provide more diagnostic information in the system log files.

Background:

The network backbone of Pleiades comprises a pair of InfiniBand fabrics that are the currently the largest in the world (for details, see Network Resources). Most of the time, the large number of switches and cables works well, but sometimes, a cable will go bad, or its connection will work loose, causing some data packets to be lost or corrupted. When one cable fails, packets get re-routed, putting additional load on other paths, which can result in congestion and dropped packets.

The Lustre and TCP/IP protocols generally handle these failures by detecting bad or missing packets and retrying. The various MPI implementations cope less well, and with different degrees of success.

So, if your MPI program aborts with an error message that suggests some node had communication problems with another node (see above) or if the program hangs after issuing such an error message, then the program might have been affected by a cable failure.

Be aware that some communication errors are not caused by bad hardware - one rank running out of memory can cause communication error messages from surviving ranks.

While we monitor for errors in the InfiniBand fabric, and replace bad or unreliable cables when we detect them, paradoxically, the act of replacing a cable can cause its own errors. Lately, there has been an increase in cables needing work, possibly as a result of the recent facility over-heating incident in early April 2011.

Resolved

Jobs Alternate Between Running and Queued States

***Problem**: Users experience unexplained job behavior, where a job alternates between Running and Queued states.*

Status: Resolved

If you are still experiencing issues with this problem, contact the NAS Control room: (800) 331-8737, (650) 604-4444, support@nas.nasa.gov.

Actions:

Updated 06.02.11 - NAS systems staff have implementing an automated method to detect processes that are running out of memory. The owner of the job will get an e-mail and the job will be terminated or blocked from rerunning.

Tips:

- If your job is bouncing between Running and Queued states in PBS, then you should assume you have an out-of-memory (OOM) situation and kill your job using the command *qdel*. You can get confirmation of the OOM situation by checking whether a job was killed by the OOM killer; or contact the NAS Control Room staff at (800) 331-8737 or (650) 604-4444.
- If you have processes running out of memory, you can increase the memory available to the processes. For example:
    - ♦ When running on Harpertown nodes, try running on Westmeres, which have twice as much memory per core.
    - ♦ When running on Westmere nodes, try running on Nehalems, which have 50% more memory per core.
    - ♦ Try running with fewer active cores in each node, and running on more nodes.
    - ♦ Run the *rank0* process in a node by itself, and add 1 to the number of nodes.

Background:

The way in which the system kills processes that are running out of memory has been changed. While the new method leaves the host node in a better state than before, the user no longer gets a message that the out-of-memory condition occurred. Furthermore, the killing is so "efficient" that PBS does not get notified. Consequently, PBS re-queues the job as if it were affected by a system problem.

In addition, SUSE Linux Enterprise 11 (SLES11) has slightly less memory available for processes than was available under SLES10. The combination means that *some* codes that ran fine with SLES10 could fail inexplicably with SLES11.

Files Fail to Open

Problem*: Users experience errors opening or inquiring about existing files using Intel Fortran on Lustre filesystems.*

Status: Resolved

If you are still experiencing issues related to this problem, contact the NAS Control room: (800) 331-8737, (650) 604-4444, support@nas.nasa.gov.

Actions:

Updated 06.14.11 - A kernel patch was installed and tested, and NAS systems staff verified that the problem no longer occurs. The patch was implemented during the Pleiades dedicated time June 8-13.

Tips:

Several workarounds were available before the kernel patch was installed.

1. Pre-load a getcwdHack library before running the executable.

   This library is available under the directory */nasa/lustre_getcwd* and was built under SLES11 SP1.

   If you plan to use this library under SLES10, you can copy the directory

   /nasa/lustre_getcwd

   to your own directory, and build it under SLES10 (bridge1 & bridge2).

   Note that no modification to your source code is needed.

   Add the following to your PBS script before running your executable.

   For csh:

   ```
   setenv LD_PRELOAD /nasa/lustre_getcwd/libgetcwdHack.so
   ```

For bash:

export LD_PRELOAD=/nasa/lustre_getcwd/libgetcwdHack.so

2. Modify your source code to re-try the file open. For example:

```
integer open_stat  (needs to be declared in this routine)

      ntries = 0   ! number of tries to open the file

 100  OPEN (UNIT=10, FILE='some_filename', STATUS='old', IOSTAT=open_stat)

      if (open_stat .ne. 0) then
         ntries = ntries + 1
         if (ntries .gt. 10) then
            print *, 'Cannot open file some_filename'
            call MPI _ABORT(MPI_COMM_WORLD, 1, ierr)
         endif
         call sleep(1)   ! to wait 1 second before retrying
         go to 100
      endif
```

3. If the file is intended to be read-only, you can change the file permission by typing "chmod 400 *filename*" or you can modify the source code to specify that the file is read-only.

```
OPEN (UNIT=10, FILE='some_filename', STATUS='old', ACTION='READ')
```

Background

With Intel Fortran and Lustre filesystems, a problem has been reported when large numbers of MPI ranks attempt to open the same file, resulting in a variety of error messages:

- forrtl: severe (9): permission to access file denied, unit xx, file /filename
- forrtl: severe (29): file not found, unit xx, file /filename
- forrtl: No such file or directory
  forrtl: severe (29): file not found, unit xx, file -/filename

In these cases, a superfluous backslash ("/") or an additional random character, such as hyphen ("-") was placed in front of the filename by the Fortran Runtime Library. This is because a *getcwd* command was issued to find the current directory and it gets "bad" information from the system. This results in the file being inaccessible or not found.

A Lustre bug report proposes a kernel patch, as well as a library built to pre-load in which *getcwd* is retried several times.